argparse Command Line Argument Parsing

argparse is a command line argument parser inspired by Python's "argparse" library. Use this with Rscript to write "#!"-shebang scripts that accept short and long flags/options and positional arguments, generate a usage statement, and set default values for options that are not specified on the command line.

In our working directory we have two example R scripts, named "example.R" and "display_file.R" illustrating the use of the argparse package.

bash$ ls

```
display_file.R
example.R
```

In order for a *nix system to recognize a "#!"-shebang line you need to mark the file executable with the `chmod` command, it also helps to add the directory containing your Rscripts to your path:

bash$ chmod ug+x display_file.R example.R

bash$ display_file.R example.R

Here is what "example.R" contains:

bash$ display_file.R example.R

```
#!/usr/bin/env Rscript
# Copyright 2012-2013 Trevor L Davis <trevor.l.davis@stanford.edu>
# Copyright 2008 Allen Day
#
#  This file is free software: you may copy, redistribute and/or modify it
#  under the terms of the GNU General Public License as published by the
#  Free Software Foundation, either version 2 of the License, or (at your
#  option) any later version.
#
#  This file is distributed in the hope that it will be useful, but
#  WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
#  General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program.  If not, see <http://www.gnu.org/licenses/>.
suppressPackageStartupMessages(library("argparse"))

# create parser object
parser <- ArgumentParser()

# specify our desired options
# by default ArgumentParser will add an help option
parser$add_argument("-v", "--verbose", action="store_true", default=TRUE,
    help="Print extra output [default]")
parser$add_argument("-q", "--quietly", action="store_false",
    dest="verbose", help="Print little output")
parser$add_argument("-c", "--count", type="integer", default=5,
    help="Number of random normals to generate [default %(default)s]",
    metavar="number")
parser$add_argument("--generator", default="rnorm",
    help = "Function to generate random deviates [default \"%(default)s\"]")
parser$add_argument("--mean", default=0, type="double",
    help="Mean if generator == \"rnorm\" [default %(default)s]")
parser$add_argument("--sd", default=1, type="double",
        metavar="standard deviation",
    help="Standard deviation if generator == \"rnorm\" [default %(default)s]")

# get command line options, if help option encountered print help and exit,
# otherwise if options not found on command line then set defaults,
args <- parser$parse_args()

# print some progress messages to stderr if "quietly" wasn't requested
if ( args$verbose ) {
    write("writing some verbose output to standard error...\n", stderr())
}

# do some operations based on user input
if( args$generator == "rnorm") {
    cat(paste(rnorm(args$count, mean=args$mean, sd=args$sd), collapse="\n"))
} else {
    cat(paste(do.call(args$generator, list(args$count)), collapse="\n"))
}
cat("\n")
```

By default *argparse* will generate a help message if it encounters `--help` or `-h` on the command line. Note how `%(default)s` in the example program was replaced by the actual default values in the help statement that *argparse* generated.

bash$ example.R --help

```
usage: example.R [-h] [-v] [-q] [-c number] [--generator GENERATOR] [--mean MEAN] [--sd standard deviation]

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         Print extra output [default]
  -q, --quietly         Print little output
  -c number, --count number
                        Number of random normals to generate [default 5]
  --generator GENERATOR
                        Function to generate random deviates [default "rnorm"]
  --mean MEAN           Mean if generator == "rnorm" [default 0]
  --sd standard deviation
                        Standard deviation if generator == "rnorm" [default 1]
```

If you specify default values when creating your `ArgumentParser` then *argparse* will use them as expected.

bash$ example.R

```
writing some verbose output to standard error...

-1.30213638533284
-0.222976011203158
-1.04143034596683
1.12333115824031
0.647999009986279
```

Or you can specify your own values.

bash$ example.R --mean=10 --sd=10 --count=3

```
writing some verbose output to standard error...

5.88309829093562
-6.59833288751021
24.8843482250535
```

If you remember from the example program that `--quiet` had `action="store_false"` and `dest="verbose"`. This means that `--quiet` is a switch that turns the `verbose` option from its default value of `TRUE` to `FALSE`. Note how the `verbose` and `quiet` options store their value in the exact same variable.

bash$ example.R --quiet -c 4 --generator="runif"

```
0.794088304974139
0.787170794326812
0.22534163785167
0.681339969392866
```

If you specify an illegal flag then emph{argparse} will print out a usage message and an error message and quit.

bash$ example.R --silent -m 5

```
usage: example.R [-h] [-v] [-q] [-c number] [--generator GENERATOR] [--mean MEAN] [--sd standard deviation]
example.R: error: unrecognized arguments: --silent -m 5
```

If you specify the same option multiple times then emph{argparse} will use the value of the last option specified.

bash$ example.R -c 100 -c 2 -c 1000 -c 7

```
writing some verbose output to standard error...

0.232229205292742
1.3082016839655
-0.160707898973026
0.768316108672477
1.37039114350624
0.387032269849774
-0.145148595558436
```

argparse can also parse positional arguments. Below we give an example program *display_file.R*, which is a program that prints out the contents of a single file (the required positional argument, not an optional argument) and which accepts the normal help option as well as an option to add line numbers to the output.

bash$ display_file.R --help

```
usage: display_file.R [-h] [-n] file

positional arguments:
  file               File to be displayed

optional arguments:
  -h, --help         show this help message and exit
  -n, --add_numbers  Print line number at the beginning of each line [default]
```

bash$ display_file.R --add_numbers display_file.R

```
1 #!/usr/bin/env Rscript
2 # Copyright 2012-2013 Trevor L Davis <trevor.l.davis@stanford.edu>
3 #
4 #  This file is free software: you may copy, redistribute and/or modify it
5 #  under the terms of the GNU General Public License as published by the
6 #  Free Software Foundation, either version 2 of the License, or (at your
7 #  option) any later version.
8 #
9 #  This file is distributed in the hope that it will be useful, but
10 #  WITHOUT ANY WARRANTY; without even the implied warranty of
11 #  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
12 #  General Public License for more details.
13 #
14 #  You should have received a copy of the GNU General Public License
15 #  along with this program.  If not, see <http://www.gnu.org/licenses/>.
16 suppressPackageStartupMessages(library("argparse"))
17
18 parser <- ArgumentParser()
19 parser$add_argument("-n", "--add_numbers", action="store_true", default=FALSE,
20     help="Print line number at the beginning of each line [default]")
21 parser$add_argument("file", nargs=1, help="File to be displayed")
22
23 args <- parser$parse_args()
24
25 file <- args$file
26
27 if( file.access(file) == -1) {
28     stop(sprintf("Specified file ( %s ) does not exist", file))
29 } else {
30     file_text <- readLines(file)
31 }
32
33 if(args$add_numbers) {
34     cat(paste(1:length(file_text), file_text), sep = "\n")
35 } else {
36     cat(file_text, sep = "\n")
37 }
```

bash$ display_file.R non_existent_file.txt

```
Error: Specified file ( non_existent_file.txt ) does not exist
Execution halted
```

bash$ display_file.R

```
usage: display_file.R [-h] [-n] file
display_file.R: error: too few arguments
```